Microprocessor Performance, Phase II
Harnessing the Transformation Hierarchy

Yale Patt

The University of Texas at Austin

ICCD

October 8, 2007

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

Up to Now (Phase I)

- ***Maintain the artificial walls between layers***
- ***Keeps the abstraction layers secure***
  - ***Makes for a better comfort zone***
- ***(Mostly) Improve the Microarchitecture***
  - ***Pipelining***
  - ***Caches***
  - ***Branch Prediction, Speculative Execution***
  - ***Out-of-order Execution***
  - ***Trace Cache***

Up to Now (Phase I)

- ***BUT, even now: Makes for a poorly utilized chip***

- ***Future process technology won’t allow it***
  - ***Too many cores (bandwidth problem)***
  - ***Too many transistors (power, energy problem)***

The Answer: Break the Layers

- *(We already have in limited cases)*
- *Pragmas in the Language*
- *The Refrigerator*
- *X + Superscalar*
- *The algorithm, the compiler, & the microarchitecture*
  - *The Alpha 21164 experiment*

Examples

- ***Compiler, Microarchitecture***
  - ***Multiple levels of cache***
  - ***Block-structured ISA***
  - ***Part by compiler, part by uarch***
  - ***Fast track, slow track***

- ***Algorithm, Compiler, Microarchitecture***
  - ***X + superscalar – the Refrigerator***
  - ***Niagara X / Pentium Y***

- ***Microarchitecture, Circuits***
  - ***Verification Hooks***
  - ***Internal fault tolerance***

Problems:

- ***Computer People work within their layers***
- ***Too few understand outside their layer***
- ***Multiple cores: people think sequential***

Microprocessor Performance, Phase II
~~*Harnessing the Transformation Hierarchy*~~
Education

Yale Patt

The University of Texas at Austin

ICCD

October 8, 2007

At least two problems

Problem 1: “Abstraction” Misunderstood

- ***Taxi to the airport***
- ***The Scheme Chip (Deeper understanding)***
- ***Sorting (choices)***
- ***Microsoft developers (Deeper understanding)***
- ***Wireless networks (Layers revisited)***

Problem 2: Thinking in Parallel is Hard

- ***Perhaps: Thinking is Hard***

- ***What if the Programmer understood shared memory, and Synchronizing Primitives***

  - ***Would it matter?***

- ***Some simple programs for freshmen***
  - ***Pipelining (aka Streaming)***
  - ***Factorial***
  - ***Parallel Search***

On Education

- ***Object-oriented FIRST does not work***
  - ***Students do not get it***
  - ***Memorizing isn’t Learning (or, Understanding)***
- ***Motivated bottom up***
  - ***Students build on what they already know***
  - ***Continually raises the level of Abstraction***
- ***Don’t be afraid to work the student hard***
  - ***Students can digest serious meat***
  - ***Students won’t complain if they are learning***
- ***No substitute for: Design, Debug, and Fix***

  …by themselves

~~We have an Education Problem~~
We have an Opportunity

- ***Top-down vs. Bottom-up***
  - ***Learning vs. Designing***

- ***Thousands of cores, Special function units***
  - ***Ability to power on/off under program control***

- ***Applications can drive Microarchitecture***
  - ***IF we can speak the same language***

- ***Algorithms, Compiler, Microarch, Circuits all talking to each other***

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons